Dans cette application de l'Unité 5 nous allons construire des programmes basés sur les graphiques.

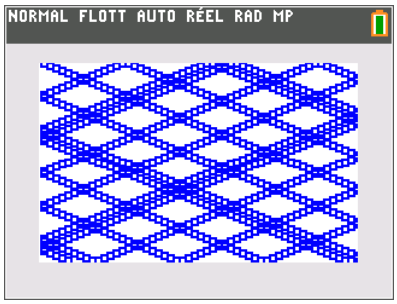
Objectifs :

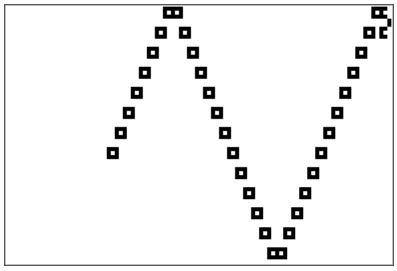
- Apprendre à détecter sur quel type de calculatrice le programme est exécuté.
- Créer un point rebondissant sur les bords de l'écran.

Indication : Ceci est un projet assez complexe qui contient des notions de physique intéressantes. Une 'particule' se déplace en ajoutant les valeurs *delta-x* et *delta-y* à ses coordonnées à chaque itération de la boucle. Ce sont les composantes horizontale et verticale de la vitesse. Lorsque la particule heurte un bord de l'écran, la composante appropriée est 'inversée' (opposée) de sorte que la particule semble 'rebondir' sur le bord de l'écran.

Programme "Pong"

Dans le jeu vidéo original 'Pong' un pixel rebondit autour de l'écran. Les joueurs contrôlaient les 'raquettes' comme au ping-pong pour garder la balle en jeu. Ce programme va reproduire le 'rebond de la balle'. Un point se déplacera suivant une ligne oblique d'un côté à l'autre de l'écran et quand il heurte le bord, il changera de direction pour sembler rebondir sur celui-ci.





Le même programme exécuté sur deux calculatrices différentes.

Le premier problème à considérer est la taille de l'écran, comme par exemple la TI-84 Plus a une taille d'écran différente de celle de la TI-83 Premium CE. Le segment de code suivant va détecter la taille de l'écran :

```
0→Xmin
1→ΔX                    également dans le menu [var]
Fenêtre…
If Xmax>95 then
  <sur une TI-83 Premium CE>
Else
  <sur une 84 Plus>
End
```

À ce stade nous connaissons maintenant le type de calculatrice et nous fixons les variables dans les blocs **Then** et **Else** pour les utiliser dans le reste du programme : **M** pour la largeur et **N** pour la hauteur.

Pour la CE : **264→M** et **165→N**

Pour la 84 Plus : **94→M** et **63→N**

Initialisation des Variables

Nous initialisons également l'étendue des *y* pour avoir de bonnes valeurs :

```
Ø→Ymin
```

1→ΔY

Nous allons faire démarrer le point **(A,B)** à un emplacement choisi de façon aléatoire, mais pas trop près du bord de l'écran :

nbrAléatEnt(10,M-10)→A
nbrAléatEnt(10,N-10)→B

Nous allons aussi initialiser les deux variables pour représenter le mouvement. Elles seront ajoutées aux coordonnées du point pour le déplacer à une nouvelle position :

nbrAléatEnt(2,5)→D changement dans A
nbrAléatEnt(2,5)→E changement dans B

Indication : Ce sont les valeurs de delta-x et delta-y.

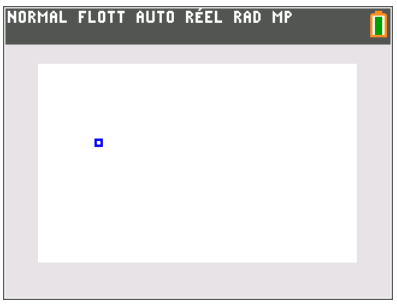
Commençons

Nous sommes prêts à démarrer l'action. Nous allons utiliser une boucle infinie :

```
While 1
  Pt-Aff(A,B,2)        le style n°2 est un grand carré

  <Reste du programme>

End
```

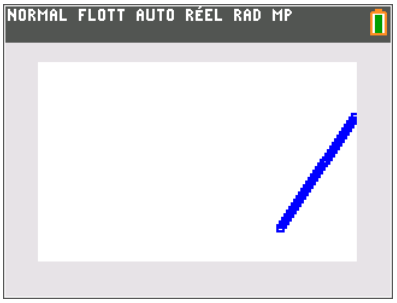


Conseil : C'est une bonne idée de mettre le **End** *en même temps afin d'en garder la trace.*

Faire bouger

Dans la boucle, après **Pt-Aff(** ajoutez le 'changement' de variables pour les coordonnées du point :

```
A+D→A
B+E→B
```



Ceci change les coordonnées du point.
Si vous exécutez le programme maintenant vous allez voir le point partir dans une direction et rapidement sortir de l'écran comme dans l'image de droite.

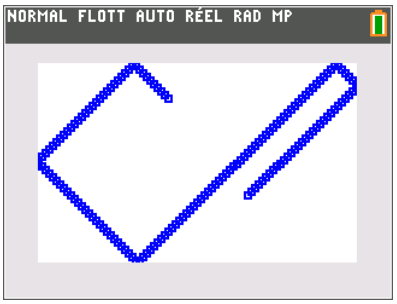
Rebond

Pour faire détecter le bord de l'écran au point nous ajoutons les instructions **If**…

```
If A>M ou A<Ø
Then

<ceci arrive lorsque le point est en dehors des limites de l'écran>

End
```



Obtenez-vous ceci ?

Deux choses doivent se produire à l'intérieur du **Then** :

- Faire revenir le point à l'écran **A-E→A**
- Changer la direction en la direction opposée **-E→E**

Écrivez une instruction If similaire pour traiter les ordonnées **B** et sa variable de ‘changement’ **E.**

Prolongements

Ne pas laisser de trace…

L’instruction **Pt-NAff** permet de cacher un point. Ajoutez une instruction **Pt-NAff** à votre programme afin que la trace du point ne soit pas affichée, seulement le point en mouvement. Attention de ne pas ‘éteindre’ le même point que l’on vient ‘d’allumer’. ‘Éteignez’ le point *précédent* (vous allez avoir besoin de deux variables de plus). Le nouveau code doit être mis à la bonne place à l’intérieur du programme.

Sortir de la boucle infinie…

getKey (menu E/S de [prgm]) donnera une valeur représentant une touche *sans* mettre le programme en pause comme **Prompt** ou **Input.** La touche [entrer] a la valeur 105 (ligne 10, colonne 5 sur le clavier).

Voici le squelette de ce qu’il faut faire :

```
Ø→K
While K≠1Ø5

   <Votre programme ici>

getKey→K
End
```

Votre tâche est de décider où ces instructions doivent se trouver dans le programme afin que l’appuie sur la touche [entrer] provoque la fin du programme.

Que pensez-vous de cela ?…

Quand vous appuyez sur [annul] (quelle valeur de **getKey** obtenez-vous ?) le programme redémarre avec un écran vide et de nouvelles valeurs aléatoires et quand vous appuyez sur [entrer] le programme se termine.

Exemple de réponse :

```
prgmPONG
FoncNAff
GraphNAff
AxesNAff
EffDess
Ø →Xmin
1 →ΔX
If Xmax>95
Then
  264 →M
  165 →N
Else
  94 →M
  63 →N
End
Ø →Ymin
1 →ΔY
nbrAléatEnt(1Ø,M – 10) →A
nbrAléatEnt(1Ø,N – 10 )→B
nbrAléatEnt(2,5) →D
nbrAléatEnt(2,5) →E

While 1
  Pt-Aff(A,B,2)
  A+D →A
  B+E →B
  If A>M or A<Ø
  Then
    A – D →A
    -D →D
  End
  If B>N or B<Ø
  Then
    B – E →B
    -E →E
  End
End
```